WGCNA Co-Expression Analysis Pipeline

Project Overview

This repository implements a **Weighted Gene Co-Expression Network Analysis (WGCNA)** pipeline for bulk RNA-seq data. WGCNA identifies clusters (modules) of highly correlated genes and relates them to external traits, enabling discovery of gene networks underlying biological processes. The pipeline automates data import, quality control, network construction, module detection, and downstream enrichment, producing reproducible results and visualizations.

Analysis Purposes

- **Module Identification**: Detect gene modules whose expression patterns are tightly co-regulated across samples.
- **Trait Association**: Correlate module eigengenes with sample-level phenotypes (e.g., disease status, clinical measurements).
- **Biological Interpretation**: Perform functional enrichment (GO/KEGG) on module genes to uncover pathways driving observed traits.
- **Data-Driven Hypotheses**: Generate testable hypotheses about key gene regulators or pathways for further experimental validation.

Key Concepts

- **Soft-Threshold Power**: A parameter that transforms gene correlation into adjacency, aiming for scale-free topology ($R^2 \geq 0.8$).
- **Topological Overlap Matrix (TOM)**: Measures interconnectedness of gene pairs, promoting robust module detection by accounting for shared neighbors.
- **Dynamic Tree Cutting**: Algorithm for identifying modules from the gene clustering dendrogram based on TOM.
- **Module Eigengene (ME)**: First principal component of gene expression within a module, summarizing module activity across samples.
- **Module–Trait Correlation**: Pearson correlation between MEs and external traits, highlighting biologically meaningful modules.
- **Functional Enrichment**: Over-representation analysis (e.g., clusterProfiler) of genes from significant modules to identify enriched GO terms or pathways.

Pipeline Modularity & Key Steps

The `wgcna_pipeline.R` script is organized into distinct sections for clarity and reusability:

1. **Data Import & Preprocessing**
   - Load expression matrix (`data/expression_matrix.tsv`) and sample traits (`data/sample_traits.tsv`).
   - Perform quality control: check for missing values, outliers, and normalize if needed (e.g., variance-stabilizing transformation via `DESeq2`).
   - Filter low-expression genes (e.g., remove genes with counts < 10 in > 90% of samples).
   - (Optional) Batch correction using `limma::removeBatchEffect()` if metadata contains a `Batch` column.
2. **Sample Clustering & Outlier Detection**
   - Compute sample-to-sample Euclidean distances and plot a dendrogram (`results/sampleClust_dendrogram.png`).
   - Identify and remove outlier samples based on clustering height threshold.
3. **Soft-Threshold Power Selection**
   - Test powers from 1 to 20 using `pickSoftThreshold()`.
   - Plot scale-free topology fit index and mean connectivity (`results/softThreshold_power.png`).
   - Choose lowest power where $R^2 \geq 0.8$ (or default to 6 if none meets criterion).
4. **Network Construction & TOM Calculation**
   - Compute adjacency matrix: `adjacency = abs(cor(exprFiltered))^power`.
   - Calculate Topological Overlap Matrix (TOM) and corresponding dissimilarity ("1 - TOM").
   - Parallelize TOM computation if `--parallel TRUE` and `WGCNA::enableWGCNAThreads()` is configured.

5. **Module Detection**
    - Hierarchical clustering of gene dissimilarity (1 - TOM) to build gene dendrogram.
    - Dynamic tree cutting (`cutreeDynamic`) with parameters: `minModuleSize` and `deepSplit` (default: 30, 2).
    - Merge modules with eigengene correlation > 0.75 using `mergeCloseModules()`.
    - Plot module-colored dendrogram (`results/module_dendrogram.png`).
6. **Eigengene Calculation & Module–Trait Correlation**
    - Compute module eigengenes (MEs) via `moduleEigengenes()`.
    - Correlate MEs with sample traits using Pearson correlation and generate a heatmap (`results/ME_trait_heatmap.png`).
    - Identify modules with |correlation| ≥ 0.5 and p-value < 0.05 as biologically relevant.
7. **Functional Enrichment of Modules**
    - For each significant module, extract gene list and run `clusterProfiler::enrichGO()` with `OrgDb = org.Hs.eg.db` (or user-specified OrgDb).
    - Save enrichment tables as `results/enrichment_<module>.tsv` and dotplots (`results/enrichment_<module>_dotplot.png`).
8. **Network Visualization for Selected Modules**
    - Export module-specific subnetworks as edge lists for use in Cytoscape or `WGCNA::plotNetwork()`.
    - Generate an interactive network plot with `igraph` or `visNetwork` (optional).
9. **RMarkdown Report Generation**
    - Compile an HTML report (`results/wgcna_report.html`) summarizing methods, parameter choices, and key figures.
    - Include session info, versioned package list for reproducibility.
10. **Logging & Session Info**

- Capture start/end time and runtime for each major step in a log file (`results/wgcna_log.txt`).
- Save `sessionInfo()` at script end to record R version and package versions.

Prerequisites

- **R ≥ 4.0**
- **Required R packages:**

```
install.packages(c("data.table", "WGCNA", "clusterProfiler", "org.Hs.eg.db", "ggplot2", "pheatmap", "optpars
BiocManager::install(c("DESeq2", "limma", "GSVA"))
```

- **Optional:**
    - **TxDb / OrgDb packages** (e.g., `TxDb.Hsapiens.UCSC.hg38.knownGene`) for gene ID conversion.
    - **SLURM / HPC environment** for parallel computing (e.g., using `parallel::mclapply` or `WGCNA::enableWGCNAThreads()`).

Input Files & Structure

1. **`data/expression_matrix.tsv`**
    - Tab-delimited file: rows = genes (Ensembl IDs or symbols), columns = samples.
    - Values: normalized expression (e.g., log2(TPM+1) or VST counts).
2. **`data/sample_traits.tsv`**
    - Tab-delimited file with header. Required columns:
        - `SampleID`: matches columns in expression matrix.
        - `Trait1`, `Trait2`, ...: Numeric or categorical traits for module association.
        - (Optional) `Batch`: Factor indicating batch for correction.
3. **`data/genes_annotation.rds`** (Optional)

- RData/RDS file mapping gene IDs to gene symbols or functional annotations.
- Used for enrichment labeling if gene IDs are not human-readable.

4. **Directory Expectations**

```
WGCNA-Coexpression-Analysis/
├── docs/
│   └── README.pdf            # Full PDF with extended documentation
├── data/
│   ├── expression_matrix.tsv
│   ├── sample_traits.tsv
│   └── genes_annotation.rds  # Optional
├── scripts/
│   └── wgcna_pipeline.R
├── results/
│   ├── sampleClust_dendrogram.png
│   ├── softThreshold_power.png
│   ├── module_dendrogram.png
│   ├── ME_trait_heatmap.png
│   ├── enrichment_<module>.tsv
│   ├── enrichment_<module>_dotplot.png
│   ├── wgcna_report.html
│   └── wgcna_log.txt
├── slurm/
│   └── run_wgcna.sbatch
└── README.md               # Quick-start version of this file
```

Quick Start

1. **Clone & Navigate**

```
git clone https://github.com/<YOUR-USERNAME>/WGCNA-Coexpression-Analysis.git
cd WGCNA-Coexpression-Analysis
```

2. **Install Dependencies**

```
install.packages(c("data.table","WGCNA","clusterProfiler","org.Hs.eg.db","ggplot2","pheatmap","optparse"))
BiocManager::install(c("DESeq2","limma","GSVA"))
```

3. **Prepare Input**

   - Place `expression_matrix.tsv` and `sample_traits.tsv` in `data/`.
   - If applicable, save gene annotation as `data/genes_annotation.rds`.

4. **Run WGCNA Pipeline**

```
Rscript scripts/wgcna_pipeline.R \
  --expr data/expression_matrix.tsv \
  --traits data/sample_traits.tsv \
  --out results/wgcna_results
```

   - This executes all modules (preprocessing → module detection → enrichment) and generates outputs in `results/`.

5. **Inspect Results**

   - **`results/sampleClust_dendrogram.png`**: Dendrogram of sample clustering.
   - **`results/softThreshold_power.png`**: Scale-free fit index vs. power.
   - **`results/module_dendrogram.png`**: Gene dendrogram with module color assignment.
   - **`results/ME_trait_heatmap.png`**: Module Eigengene–Trait correlation heatmap.
   - **`results/enrichment_<module>.tsv`** & **`results/enrichment_<module>_dotplot.png`**: Functional enrichment for selected modules.
   - **`results/wgcna_report.html`**: Comprehensive HTML report summarizing all steps and figures.

Usage Notes & Customization

- **Soft-Threshold Power Override**: Use `--power <value>` to skip automatic selection.
- **Module Detection Parameters**: Customize `minModuleSize` and `deepSplit` within `scripts/wgcna_pipeline.R` or via command-line

flags (e.g., `--minModuleSize 20 --deepSplit 1`).

- **Batch Correction**: If `sample_traits.tsv` contains `Batch`, uncomment batch correction lines in the script to remove batch effects.
- **Custom Enrichment Databases**: Replace `org.Hs.eg.db` with other OrgDb (e.g., `org.Mm.eg.db` for mouse).
- **Output Structure**: Use `--out` prefix to direct files to a custom folder. Defaults to timestamped `results/wgcna_<timestamp>/`.
- **SLURM Integration**: Submit `slurm/run_wgcna.sbatch` on an HPC cluster, adjusting SBATCH directives for cores and memory.

Contact & License

- **Author**: Sally L. Yepes Torres
- **Email**: sallyepes233@gmail.com

- **Last Updated**: June 2025
- **License**: MIT

Please cite the WGCNA R package (Langfelder & Horvath, 2008) when using this pipeline in publications.